This paper was accepted for publication in *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING*

Model and Heuristic Solutions for the Multiple Double-Load Crane Scheduling Problem in Slab Yards

Guodong Zhao, Jiyin Liu, Lixin Tang, Senior Member, IEEE, Ren Zhao, and Yun Dong

***Abstract*—This paper studies a multiple double-load crane scheduling problem in steel slab yards. Consideration of multiple cranes and their double-load capability makes the scheduling problem more complex. This problem has not been studied previously. We first formulate the problem as a mixed-integer linear programming (MILP) model. A two-phase model-based heuristic is then proposed. To solve large problems, a pointer-based discrete differential evolution (PDDE) algorithm was developed with a dynamic programming algorithm embedded to solve the one-crane sub-problem for a fixed sequence of tasks. Instances of real problems are collected from a steel company to test the performance of the solution methods. The experiment results show that the model can solve small problems optimally, and the solution greatly improves the schedule currently used in practice. The two-phase heuristic generates near-optimal solutions, but it can still only solve comparatively modest problems within reasonable (4-hour) computational timeframes. The PDDE algorithm can solve large practical problems relatively quickly, and provides better results than the two-phase heuristic solution, demonstrating its effectiveness and efficiency and therefore its suitability for practical use.**

***Note to Practitioners*—Bridge cranes are commonly used to move heavy items in manufacturing and logistics systems. Usually, more than one crane runs on a common track. The latest versions of such cranes, such as those used in slab yards in the steel industry, can hold two items simultaneously. Operations scheduling of multiple double-load cranes involves assignment of tasks to the cranes, combination of tasks to double-load operations, and sequencing of the tasks, considering the non-crossing constraint between cranes. Effective solution of this complex problem can help fully utilize the crane capability, increase productivity, and reduce energy consumption. This paper models this problem and develops a heuristic solution that combines differential evolution and dynamic programming. Experiment results show that the algorithm is effective and efficient for practical use in slab yards. It may also be applicable to other systems using similar cranes.**

***Index Terms*—Multiple crane scheduling, Double-load cranes, Integer programming, Differential evolution, Dynamic programming.**

I. Introduction

STEEL slabs are intermediate products in the steel production system. They are produced in the continuous-casting stage and are used as materials in the hot-rolling stage. To ensure smooth production, slabs are stacked in a storage yard/warehouse that functions as a buffer between the two stages. Based on production requirement, the slabs may need to be moved between two stacks, or between a stack and an entry or exit point. These movement tasks are carried out by bridge cranes. The warehouse may consist of several halls, but their operations are independent of each other and so can be planned separately. There can be two or more cranes in the same hall, running on a common overhead rail track. Fig. 1 shows the layout of an example storage hall with two cranes.

The hall is often divided into several areas in practice, such that each area is served by one crane. With this practical strategy, a single-crane scheduling problem can be solved to optimize the operations of each storage area. The most advanced cranes can hold up to two slabs simultaneously. The problem of scheduling such double-load cranes is more complex than scheduling the traditional cranes which hold only one slab at a time. Zhao et al. [1] studied the problem of scheduling a single crane with double-load capability.

However, cranes in the same hall can travel to different areas. In situations where the workloads are not balanced, higher productivity may be achieved if cranes are allowed to assist each other rather than being restricted to their own areas. As the cranes run on the same track, they cannot cross each other and must also maintain a safety distance. When more than one crane is considered, there are also new decisions on the assignment of tasks to cranes. These new constraints and decisions add additional complexity to the problem. In this paper we study this problem of scheduling multiple cranes each with double-load capability.

Problems of scheduling multiple cranes on a common track have been studied previously in different contexts. Lieberman and Turksen [2] studied a crane scheduling problem in production systems, and proposed heuristic solutions considering situations where the tasks are ready at the same

This work was supported by the Fund for Innovative Research Groups of the National Natural Science Foundation of China (71621061), the Major International Joint Research Project of the National Natural Science Foundation of China (71520107004), General fund of the National Natural Science Foundation of China (71472081), and the 111 Project (B16009).

Guodong Zhao is with Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, Liaoning Engineering Laboratory of Data Analytics and Optimization for Smart Industry, Shenyang, 110819, China. (E-mail: gascoigne_0509@163.com).

Jiyin Liu is with the School of Business and Economics, Loughborough University, Leicestershire LE11 3TU, U.K. (E-mail: j.y.liu@lboro.ac.uk).

Lixin Tang is with Institute of Industrial and Systems Engineering, Liaoning Key Laboratory of Manufacturing System and Logistics. Northeastern University, Shenyang, 110819, China. (E-mail: lixintang@ise.neu.edu.cn).

Ren Zhao and Yun Dong are with Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, Institute of Industrial and Systems Engineering, Northeastern University, Shenyang, 110819, China. (E-mails: zhaoren@ise.neu.edu.cn; dydexter@hotmail.com).

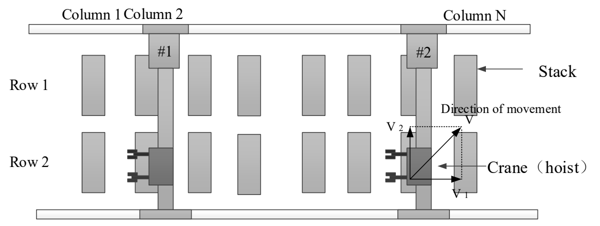

Fig. 1. An example slab storage hall with two cranes.

time and at different times respectively. More research was done on crane scheduling problems in container terminal operations, mostly at container ports. These include yard crane scheduling problems (e.g., Lim et al. [3]; Ng [4]; Li et al. [5]; Li et al. [6]), quay crane scheduling problems (e.g., Liu et al. [7]; Lee et al. [8]; Chen et al. [9]; Alosufi et al. [10]), and combined problems (e.g., Zhang et al. [11]). Guo et al. [12] studied a crane scheduling problem in a railroad container terminal. Zheng et al. [13] studied a crane scheduling problem with uncertain release times of retrieval tasks in a container yard. They proposed a two-stage stochastic programming model and developed a heuristic algorithm to solve the problem. The tasks in our problem are all ready to perform though there may be precedence constraints among them. If anticipated future tasks are also included in the problem, then the method in the above reference could be useful for handling possible uncertainties in the task arrivals. In all these problems each task needs to be performed by a crane at one fixed position, while the tasks in the problem studied in this paper involve a crane movement from one position to another.

Crane scheduling problems with movement tasks are often seen in production systems. Hirsch et al. [14] studied a two-crane routing problem in the roof-tile industry, where the tasks involved moving workpieces on iron pallets from stage to stage and moving the pallets back in a four-stage hybrid flowshop system. An ant colony optimization (ACO)-based solution approach was used to solve the problem. Other studies examined cyclic production in electroplating lines, where items for plating are moved between the line stages by hoists on a common track. Lei and Wang [15] studied a problem with two hoists where part movements were all in one direction. The line was partitioned into two non-overlapping zones and the tasks in each zone were assigned to one hoist. Che and Chu [16] proposed a branch-and-bound algorithm for a multi-hoist problem with all part movements in one direction. Zhou and Liu [17] proposed a heuristic search algorithm for the two-hoist problem allowing movements in both directions. Leung and Zhang [18] and Leung et al. [19] presented mixed-integer linear programming (MILP) models for the multi-hoist scheduling problem. Liu and Jiang [20] and Jiang and Liu [21] analyzed a no-wait version with fixed processing times for multi-hoist scheduling problems and developed polynomial time algorithms to solve them optimally. Based on the analysis, a new MILP model is developed for the general problem, which enables an efficient branch-and-bound solution (Jiang and Liu, [22]).

There has been some previous research studying crane scheduling problems in the steel industry. Dohn and Clausen [23] studied a slab yard planning problem to decide the slab movements, and a crane scheduling problem to carry out the movement tasks. The crane scheduling problem was solved using a greedy heuristic and local search. Xie et al. [24] studied a multi-crane scheduling problem arising in the batch annealing process, to move heaters and coolers among the steel coils stacked on annealing bases. A MILP was formulated, and a heuristic solution algorithm was proposed with analysis of its worst-case performance. Maschietto et al. [25] studied a two-crane scheduling problem in a steel coil distribution center and solved it using a genetic algorithm. Zhou *et al.* [26] studied a new combined transportation problem for China's "the Belt and Road" initiative, they proposed a time–distance-based cost to compare with traditional transportation. The research considered the combination of sea–rail transportation, while our problem is to consider the combination of different tasks.

In all of the above studies, the cranes or hoists can only handle one item. In this paper, the crane can hold two slabs simultaneously, and so two slab-moving tasks can be combined in one loaded trip. Considering this new feature, we need to decide both the task combination and the crane routes to perform all the tasks.

In the following sections of the paper, we first describe the problem in more detail and present a MILP model (Section II). A two-phase heuristic is proposed in Section III. Section IV combines differential evolution and dynamic programming to solve the problem. Section V reports experiment results. Conclusions are drawn in Section VI.

II. Problem Description and Modeling

Slabs are stored in stacks in the warehouse hall. These stacks are arranged in rows and columns as shown in Fig. 1. The position of a stack can be indicated by its row and column numbers. There are *m* cranes in the hall. When a crane moves a slab from one stack to another, the crane bridge travels along the length of the hall and the hoist travels on the bridge along the width of the hall. The time needed for the movement is the longer one of these two travel times. In practice, the width of the hall is much shorter and so the travel time is determined by the distance traveled in the length direction. Therefore, it is sufficient to just use the column position to represent the stack position and only consider the crane movements in one dimension. An entry or exit point can be viewed as a stack. All the cranes run on a common track. Although each of them can travel along the full length of the hall, they cannot cross each other, and any two adjacent cranes must also maintain a minimum safety distance.

In the problem studied, a set of slab-movement tasks are given. These may include receiving and storage, rearrangement, retrieval, and shuffling. Each task requires the movement of one slab from its initial stack to its target stack. There may be precedency requirements among the tasks. For example, priority must to be given to urgent tasks; the task of picking up a slab in a higher position must be performed before the task of picking up a slab underneath it.

A crane can hold at most two slabs simultaneously. Therefore, it can perform single tasks one after another, or combine two tasks in a double-load operation. In a double-load operation the crane picks up the first slab from its initial stack, travels to the initial stack of the second slab, picks up the second slabs, travels to the target stack of the second slab, drops off the second slab, then travels to the target stack of the first slab and drops it off there. Some of these steps may be saved if the two slabs have the same initial stack and/or the same target stack. No more than two slabs are allowed to be combined in a double-load operation, i.e., the crane cannot pick up another slab before both slabs are dropped off. Note that a double-load operation combining task A and task B is different from that combining B and A. To combine two tasks, the slab of the first task cannot be wider than that of the second task. Slab widths are known for the given tasks.

Considering the tasks and constraints described above, the scheduling problem is to allocate the tasks to the cranes and determine the schedule for the cranes to perform the tasks, including the combination of tasks into double-load operations, so as to complete all the tasks as soon as possible, i.e., to minimize the makespan.

A. Parameters

O	set of tasks, $O = \{1, \ldots, N\}$.
K	set of cranes, $K = \{1, \ldots, m\}$.
o_i^+	initial stack of task *i*.
o_i^-	target stack of task *i*.
v	maximum crane speed when carrying one slab.
v^+	maximum crane speed when carrying two slabs.
v^-	maximum crane speed when empty.
$p_{i+,j-}$	1 if task *j* must complete later than task *i* starts, otherwise 0.
$p_{i+,j+}$	1 if task *j* must start later than task *i* starts, otherwise 0.
$p_{i-,j-}$	1 if task *j* must complete later than task *i* completes, otherwise 0.
$p_{i-,j+}$	1 if task *j* must start later than task *i* completes, otherwise 0.
q_{ij}	1 if task *i* and *j* meet the double-load conditions, otherwise 0.
$d_{\min}$	the safety distance between cranes.
M	A sufficiently large constant.

B. Decision Variables

$$x_{i+,j+} = \begin{cases} 1, & \text{if task } j \text{ starts later than task } i \text{ starts,} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i+,j-} = \begin{cases} 1, & \text{if task } j \text{ completes later than task } i \text{ starts,} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i-,j+} = \begin{cases} 1, & \text{if task } j \text{ starts later than task } i \text{ completes,} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i-,j-} = \begin{cases} 1, & \text{if task } j \text{ completes later than task } i \text{ completes,} \\ 0, & \text{otherwise} \end{cases}$$

$$z_i^k = \begin{cases} 1, & \text{if task } i \text{ is assigned to crane } k, \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1, & \text{if tasks } i \text{ and } j \text{ are performed together} \\ & \text{by a crane in a double-load operation,} \\ 0, & \text{otherweise} \end{cases}$$

S_i	start time of task *i*.
C_i	completion time of task *i*.
$C_{\max}$	time when all tasks are completed.
p_{i+}^k	position of crane *k* at the time point when task *i* starts.
p_{i-}^k	position of crane *k* at the time point when task *i* completes.

C. Mathematical Formulation

Using the above notation, our crane scheduling problem can be formulated as the following integer programming model.

Minimize $C_{\max}$

$$\sum_{k=1}^{m} z_i^k = 1, \quad i = 1,\ldots,N. \tag{1}$$

$$x_{i+,j+} + x_{j+,i+} = 1, \quad i,j = 1,\ldots,N,\ i \neq j. \tag{2}$$

$$x_{i+,j-} + x_{j-,i+} = 1, \quad i,j = 1,\ldots,N,\ i \neq j. \tag{3}$$

$$x_{i-,j+} + x_{j+,i-} = 1, \quad i,j = 1,\ldots,N,\ i \neq j. \tag{4}$$

$$x_{i-,j-} + x_{j-,i-} = 1, \quad i,j = 1,\ldots,N,\ i \neq j. \tag{5}$$

$$4y_{ij} \le x_{i+,j+} + x_{j-,i-} + z_i^k + z_j^k, \quad k = 1,\ldots,m,\ i,j = 1,\ldots,N,\ i \neq j. \tag{6}$$

$$y_{ij} \ge x_{i+,j+} + x_{j-,i-} + z_i^k + z_j^k - 3, \quad k = 1,\ldots,m,\ i,j = 1,\ldots,N,\ i \neq j. \tag{7}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{N} y_{ij} + \sum_{\substack{l=1 \\ l \neq j}}^{N} y_{jl} \le 1, \quad j = 1,\ldots,N. \tag{8}$$

$$y_{ij} \le q_{ij}, \quad i,j = 1,\ldots,N,\ i \neq j. \tag{9}$$

$$p_{i+}^k \le o_i^+ + M(1 - z_i^k), \quad k = 1,\ldots,m,\ i = 1,\ldots,N. \tag{10}$$

$$p_{i+}^k \ge o_i^+ + M(z_i^k - 1), \quad k = 1,\ldots,m,\ i = 1,\ldots,N. \tag{11}$$

$$p_{i-}^k \le o_i^- + M(1 - z_i^k), \quad k = 1,\ldots,m,\ i = 1,\ldots,N. \tag{12}$$

$$p_{i-}^k \ge o_i^- + M(z_i^k - 1), \quad k = 1,\ldots,m,\ i = 1,\ldots,N. \tag{13}$$

$$p_{i+}^k + d_{\min} \le p_{i+}^{k+1}, \quad k = 1,\ldots,m-1,\ i = 1,\ldots,N. \tag{14}$$

$$p_{i-}^k + d_{\min} \le p_{i-}^{k+1}, \quad k = 1,\ldots,m-1,\ i = 1,\ldots,N. \tag{15}$$

$$|p_{i+}^k - p_{i-}^k|/v \le (C_i - S_i) + M\Big(1 - z_i^k + \sum_{\substack{j=1 \\ j \neq i}}^{N} y_{ji} + \sum_{\substack{j=1 \\ j \neq i}}^{N} y_{ij}\Big), \quad k = 1,\ldots,m,\ i = 1,\ldots,N. \tag{16}$$

$$|p_{i+}^k - p_{j+}^k|/v \le (S_i - S_j) + M(2 - z_i^k - y_{ji}), \quad k = 1,\ldots,m,\ i,j = 1,\ldots,N,\ i \neq j. \tag{17}$$

$$|p_{i-}^k - p_{j-}^k|/v \le (C_j - C_i) + M(2 - z_i^k - y_{ji}), \quad k = 1,\ldots,m,\ i,j = 1,\ldots,N,\ i \neq j. \tag{18}$$

$$| p_{i+}^k - p_{i-}^k | / v^+ \le (C_i - S_i) + M(2 - z_i^k - \sum_{j \in O, j \ne i} y_{ji}), \quad k = 1,...,m,\ i = 1,...,N. \tag{19}$$

$$| p_{i+}^k - p_{j+}^k | / v^- \le (S_j - S_i) + M(1 - x_{i+,j+}), \quad k = 1,...,m,\ i,j = 1,...,N,\ i \ne j. \tag{20}$$

$$| p_{i+}^k - p_{j-}^k | / v^- \le (C_j - S_i) + M(1 - x_{i+,j-}), \quad k = 1,...,m,\ i,j = 1,...,N,\ i \ne j. \tag{21}$$

$$| p_{i-}^k - p_{j+}^k | / v^- \le (S_j - C_i) + M(1 - x_{i-,j+}), \quad k = 1,...,m,\ i,j = 1,...,N,\ i \ne j. \tag{22}$$

$$| p_{i-}^k - p_{j-}^k | / v^- \le (C_j - C_i) + M(1 - x_{i-,j-}), \quad k = 1,...,m,\ i,j = 1,...,N,\ i \ne j. \tag{23}$$

$$S_j - S_i \ge d_{\min} / v, \qquad p_{i+,j+} = 1, i,j = 1,...,N,\ i \ne j. \tag{24}$$

$$C_j - S_i \ge d_{\min} / v, \qquad p_{i+,j-} = 1, i,j = 1,...,N,\ i \ne j. \tag{25}$$

$$S_j - C_i \ge d_{\min} / v, \qquad p_{i-,j+} = 1, i,j = 1,...,N,\ i \ne j. \tag{26}$$

$$C_j - C_i \ge d_{\min} / v, \qquad p_{i-,j-} = 1, i,j = 1,...,N,\ i \ne j. \tag{27}$$

$$C_i \le C_{\max}, \qquad i = 1,...,N. \tag{28}$$

$$x_{i+,j+}, x_{i+,j-}, x_{i-,j+}, x_{i-,j-}, y_{ij}, z_i^k \in \{0,1\}, \quad i,j = 1,...,N,\ i \ne j,\ k = 1,...,m. \tag{29}$$

$$p_{i+}^k, p_{i-}^k, S_i, C_i \ge 0, \qquad i = 1,...,N,\ k = 1,...,m. \tag{30}$$

In this model, the objective is to minimize the time required to complete all the crane tasks. Constraints (1) ensure that each task is performed by one crane. Constraints (2) to (5) ensure that any two events (start and completion time points of tasks) can happen in only one order. Constraints (6) and (7) require that the two tasks in a double-load operation must be performed by the same crane. Constraints (8) mean that any task may be carried out in a double-load operation with at most one other task. Constraints (9) indicate that the double-load conditions must be met if two tasks are combined in a double-load operation. Constraints (10) to (13) link the positions of cranes to those of tasks: if a crane *k* performs a task *i*, constraints (10) and (11) ensure that crane *k* must be at the initial stack of task *i* when the task starts, while constraints (12) and (13) ensure that the crane must be at the target stack of the task when the task completes. If crane *k* is not assigned to perform task *i*, then its positions at the start and completion times of this task will be determined by the model and not restricted by the positions of this task. Constraints (14) and (15) guarantee the minimum safety distance between any two adjacent cranes at all times. These two constraints also ensure that the cranes will never cross each other, as any crane *k* is required to be always on one side of crane *k*+1 and to maintain at least the safety distance. Constraints (16) to (23) require that there are sufficient times for the cranes to perform the tasks and also to make the necessary empty movements between tasks. Constraints (16) are for single-load tasks, constraints (17) to (19) are for double-load tasks, and constraints (20) to (23) are for empty crane movements. Constraints (24) to (27) ensure that the precedence requirements are satisfied. Constraints (28) states that the completion time of all tasks cannot be earlier than the completion time of any of the individual tasks. Constraints (29) and (30) define the range of variable values.

The value of *M* in the constraints needs to be sufficiently large to ensure model correctness. However, too large value of *M* may affect the efficiency of the solution process. The value of *M* can be set differently in different constraints. For constraints (10) to (13), it is sufficient to set the value of *M* to the largest distance between any two stacks. For constraints (16) to (23), the value of *M* needs to be greater than the makespan. We set it to the time required for one crane to perform all tasks one by one.

We tested the model by running it on some small problem instances. CPLEX 12.51 was used to solve the model with the maximum running time set to 14400 seconds (4 hours). The solution time increases quickly with the number of tasks in the problem, and when the number of tasks reaches 9, some instances cannot be solved optimally within the given time. If the problem size increases any further, the model cannot generate even a feasible solution within the 4-hour limit.

III. A TWO-PHASE DECOMPOSITION HEURISTIC

As an attempt to solve larger problems we try to decompose the problem into two phases: the first phase determines the task combinations assuming that all tasks are performed by a single crane; then, with the task combinations fixed, the second phase decides the task assignment and schedule. Each phase can also be formulated as a smaller MILP model.

A. Phase 1: Solve the problem as if it is a single-crane problem.

This phase assumes that there is only one crane. Therefore, the *z* variables and safety distance constraints are not needed. Some of the other constraints may also be simplified accordingly. Using the notation defined in the previous section, the model for this phase can be represented as follows.

Minimize $C_{\max}$

$$x_{i+,j+} + x_{j+,i+} = 1, \qquad i,j = 1,...,N,\ i \ne j. \tag{31}$$

$$x_{i+,j-} + x_{j-,i+} = 1, \qquad i,j = 1,...,N,\ i \ne j. \tag{32}$$

$$x_{i-,j+} + x_{j+,i-} = 1, \qquad i,j = 1,...,N,\ i \ne j. \tag{33}$$

$$x_{i-,j-} + x_{j-,i-} = 1, \qquad i,j = 1,...,N,\ i \ne j. \tag{34}$$

$$2y_{ij} \le x_{i+,j+} + x_{j-,i-}, \qquad i,j = 1,...,N,\ i \ne j. \tag{35}$$

$$y_{ij} \ge x_{i+,j+} + x_{j-,i-} - 1, \quad i,j = 1,...,N,\ i \ne j. \tag{36}$$

$$\sum_{\substack{i=1 \\ i \ne j}}^{N} y_{ij} + \sum_{\substack{l=1 \\ l \ne j}}^{N} y_{jl} \le 1, \qquad j = 1,...,N. \tag{37}$$

$$y_{ij} \le q_{ij}, \qquad i,j = 1,...,N,\ i \ne j. \tag{38}$$

$$| o_{i+} - o_{i-} | / v \le (C_i - S_i) + M(\sum_{\substack{j=1 \\ j \ne i}}^{N} y_{ji} + \sum_{\substack{j=1 \\ j \ne i}}^{N} y_{ij}), \quad i = 1,...,N. \tag{39}$$

$$| o_{i+} - o_{j+} | / v \le (S_i - S_j) + M(1 - y_{ji}), \quad i,j = 1,...,N,\ i \ne j. \tag{40}$$

$$|o_{i+} - o_{i-}|/v^{+} \le (C_i - S_i) + M(1 - \sum_{j\in O, j\ne i} y_{ji}),$$
$$i = 1,\dots,N. \tag{41}$$

$$|o_{i-} - o_{j-}|/v \le (C_j - C_i) + M(1 - y_{ji}),$$
$$i, j = 1,\dots,N,\ i \ne j. \tag{42}$$

$$|o_{i-} - o_{j+}|/v^{-} \le (S_j - C_i) + M(1 - x_{i-,j+}),$$
$$i, j = 1,\dots,N,\ i \ne j. \tag{43}$$

$$S_j - S_i \ge 0, \qquad p_{i+,j+} = 1,\ i, j = 1,\dots,N,\ i \ne j. \tag{44}$$

$$C_j - S_i \ge 0, \qquad p_{i+,j-} = 1,\ i, j = 1,\dots,N,\ i \ne j. \tag{45}$$

$$S_j - C_i \ge 0, \qquad p_{i-,j+} = 1,\ i, j = 1,\dots,N,\ i \ne j. \tag{46}$$

$$C_j - C_i \ge 0, \qquad p_{i-,j-} = 1,\ i, j = 1,\dots,N,\ i \ne j. \tag{47}$$

$$C_i \le C_{\max}, \qquad i = 1,\dots,N. \tag{48}$$

$$x_{i+,j+}, x_{i+,j-}, x_{i-,j+}, x_{i-,j-}, y_{ij}, z_i^k \in \{0,1\},$$
$$i, j = 1,\dots,N,\ i \ne j. \tag{49}$$

$$S_i, C_i \ge 0, \qquad i = 1,\dots,N. \tag{50}$$

The meanings of the constraints in this model are similar to the corresponding ones in the overall MILP model. Note that the assignment constraints and the large number of collision-avoidance constraints do not appear in this model, making it easier to solve.

After solving this first phase model, any two tasks that are performed in the same double-load operation in the solution will be viewed as one new task approximately in the second phase. Each of the tasks that are performed in single-load operations is also considered as a new task. For each new task representing a double-load operation, the "initial" and "target" positions and the duration can be calculated as follows:

Suppose that the task picked up first in this double-load operation (the new task *i*) is task A, and the other task is task B. The "initial" position o_{i+}^{n} and "target" position o_{i-}^{n} of the "new task" can be set based on the direction of task A, i.e., the relationship between its initial and target positions:

If $o_{A+} \le o_{A-}$,

$$o_{i+}^{n} = \min\{o_{A+}, o_{A-}, o_{B+}, o_{B-}\},$$

$$o_{i-}^{n} = \max\{o_{A+}, o_{A-}, o_{B+}, o_{B-}\};$$

Otherwise,

$$o_{i+}^{n} = \max\{o_{A+}, o_{A-}, o_{B+}, o_{B-}\},$$

$$o_{i-}^{n} = \min\{o_{A+}, o_{A-}, o_{B+}, o_{B-}\}.$$

The duration of the "new task":

$$t_i = |o_{i+}^{n} - o_{A+}|/v^{-} + |o_{A+} - o_{B+}|/v + |o_{B+} - o_{B-}|/v^{+} +$$
$$|o_{B-} - o_{A-}|/v + |o_{A-} - o_{i-}^{n}|/v^{-}.$$

For example, Fig. 2 shows the initial and target positions of two tasks (A and B) of a double-load operation. The initial position of A is on the left of its target position ($o_{A+} \le o_{A-}$). According to the above formulae, the "initial" and "target" positions of the "new task" will be positions 2 and 9, respectively. The duration of the "new task" is the sum of empty travel time from position 2 to position 4, single-load travel time from position 4 to position 2, double-load travel time from position 2 to position 7, and single-load travel time from position 7 to position 9.

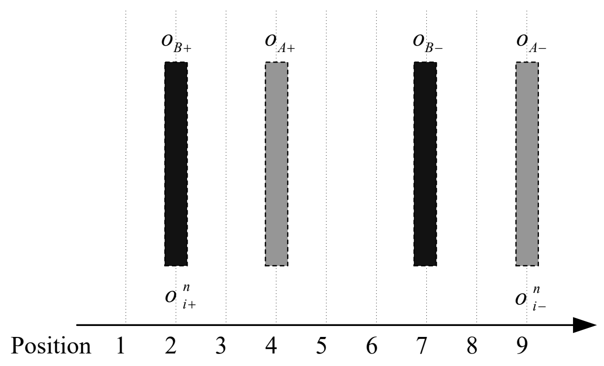

Fig. 2. Schematic of the "new task" definition

B. Phase 2: Allocate the new tasks to cranes and schedule the tasks for each crane.

The phase-two problem is then to assign the cranes to perform the new tasks without further task combination. We redefine the following parameters and variables to present the phase-two model. The parameters concerning cranes remain unchanged.

Parameters

O	set of new tasks, $O = \{1, \dots, N\}$. For simplicity, we refer to these as "tasks" when describing this model.
o_{i+}^{n}	initial position of task *i*.
o_{i-}^{n}	target position of task *i*.
t_i	duration of task *i*.
$p_{i+,j-}$	1 if task *j* must complete later than task *i* starts, otherwise 0.
$p_{i+,j+}$	1 if task *j* must start later than task *i* starts, otherwise 0.
$p_{i-,j-}$	1 if task *j* must complete later than task *i* completes, otherwise 0.
$p_{i-,j+}$	1 if task *j* must start later than task *i* completes, otherwise 0.
M	a sufficiently large constant.

Decision Variables

$$x_{i+,j+} = \begin{cases} 1, & \text{if task } j \text{ starts later than task } i \text{ starts,} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i+,j-} = \begin{cases} 1, & \text{if task } j \text{ completes later than task } i \text{ starts,} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i-,j+} = \begin{cases} 1, & \text{if task } j \text{ starts later than task } i \text{ completes,} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i-,j-} = \begin{cases} 1, & \text{if task } j \text{ completes later than task } i \text{ completes,} \\ 0, & \text{otherwise} \end{cases}$$

$z_i^k = \begin{cases} 1, & \text{if task } i \text{ is assigned to crane } k, \\ 0, & \text{otherwise} \end{cases}$

S_i start time of task *i*.

C_i completion time of task *i*.

$C_{\max}$ time when all tasks are completed.

p_{i+}^k position of crane *k* at the time point when task *i* starts.

p_{i-}^k position of crane *k* at the time point when task *i* completes.

Using the parameters and variables, the following integer programming model is established:

Minimize $C_{\max}$

$$\sum_{k=1}^{m} z_i^k = 1, \qquad i = 1,...,N. \tag{51}$$

$$x_{i+,j+} + x_{j+,i+} = 1, \qquad i,j = 1,...,N, \ i \neq j. \tag{52}$$

$$x_{i+,j-} + x_{j-,i+} = 1, \qquad i,j = 1,...,N, \ i \neq j. \tag{53}$$

$$x_{i-,j+} + x_{j+,i-} = 1, \qquad i,j = 1,...,N, \ i \neq j. \tag{54}$$

$$x_{i-,j-} + x_{j-,i-} = 1, \qquad i,j = 1,...,N, \ i \neq j. \tag{55}$$

$$\max\{o_{i+}^n, o_{i-}^n\} + d_{\min} \le \min\{o_{j+}^n, o_{j-}^n\} + M(4 - z_i^k - z_j^{k+1} - x_{i+,j+} - x_{j+,i-}), \quad k = 1,\ldots,m-1, \ i,j = 1,...,N, \ i < j. \tag{56}$$

$$\max\{o_{i+}^n, o_{i-}^n\} + d_{\min} \le \min\{o_{j+}^n, o_{j-}^n\} + M(4 - z_i^k - z_j^{k+1} - x_{i+,j-} - x_{j-,i-}), \quad k = 1,\ldots,m-1, \ i,j = 1,...,N, \ i < j. \tag{57}$$

$$\max\{o_{i+}^n, o_{i-}^n\} + d_{\min} \le \min\{o_{j+}^n, o_{j-}^n\} + M(4 - z_i^k - z_j^{k+1} - x_{j+,i+} - x_{i+,j-}), \quad k = 1,\ldots,m-1, \ i,j = 1,...,N, \ i < j. \tag{58}$$

$$S_i + t_i \le C_i, \qquad i = 1,...,N. \tag{59}$$

$$S_i \le S_j + M(1 - x_{i+,j+}), \qquad i,j = 1,...,N, \ i < j. \tag{60}$$

$$S_i \le C_j + M(1 - x_{i+,j-}), \qquad i,j = 1,...,N, \ i < j. \tag{61}$$

$$C_i \le S_j + M(1 - x_{i-,j+}), \qquad i,j = 1,...,N, \ i < j. \tag{62}$$

$$C_i \le C_j + M(1 - x_{i-,j-}), \qquad i,j = 1,...,N, \ i < j. \tag{63}$$

$$C_i + |p_{i-}^k - p_{j+}^k| / v^- \le S_j + M(3 - z_i^k - z_j^k - x_{i-,j+}), \quad k = 1,...,m, i,j = 1,...,N, \ i < j. \tag{64}$$

$$S_j - S_i \ge d_{\min} / v, \qquad p_{i+,j+} = 1, i,j = 1,...,N, \ i < j. \tag{65}$$

$$C_j - S_i \ge d_{\min} / v, \qquad p_{i+,j-} = 1, \ i,j = 1,...,N, \ i < j. \tag{66}$$

$$S_j - C_i \ge d_{\min} / v, \qquad p_{i-,j+} = 1, \ i,j = 1,...,N, \ i < j. \tag{67}$$

$$C_j - C_i \ge d_{\min} / v, \qquad p_{i-,j-} = 1, \ i,j = 1,...,N, \ i < j. \tag{68}$$

$$C_i \le C_{\max}, \qquad i = 1,...,N. \tag{69}$$

$$x_{i+,j+}, x_{i+,j-}, x_{i-,j+}, x_{i-,j-}, z_i^k \in \{0,1\}, \quad k = 1,...,m, i,j = 1,...,N, \ i < j. \tag{70}$$

$$S_i, C_i \ge 0, \qquad i = 1,...,N. \tag{71}$$

This is a multi-crane model, and the constraints are similar to those in the overall MILP model. However, all tasks in this model are "single-load" tasks and so there are no variables and constraints related to decisions on combining tasks to double-load operations. This model is thus smaller than the overall model.

This two-phase heuristic was also tested using a maximum allowable running time of 2 hours for each problem instance that was used for testing the overall MILP model. The first-phase model can reach optimal solutions relatively quickly, and all the remaining time is used to solve the second-phase model. For the instances that are solved optimally by the overall MILP model, the two-phase heuristic solution is about 5.2% to optimal on average, and its solution time is about half that of the overall MILP model. With such time performance, this heuristic is still not suitable for solving large problems, and therefore a more efficient method is needed.

IV. Heuristic solution with differential evolution

The problems that can be solved by the two-phase heuristic within a reasonable time limit are still relatively small. To solve larger practical problems, we turn to metaheuristics. Differential evolution (DE) is one of the latest metaheuristics and has shown excellent performance in solving optimization problems. It was initially proposed for solving continuous optimization problems. Subsequent modifications have made it applicable to discrete optimization problems. Tang et al. [27] studied an improved DE for steelmaking-continuous casting production. A new mutation strategy and an incremental mechanism are proposed, which can prove the efficiency and effectiveness of DE for solving scheduling problems in the steel industry. Tang et al. [28] studied differential evolution with an individual-dependent mechanism. They proposed a new parameter setting and mutation strategy for DE, which significantly improves the computational efficiency of DE. Dong and Zhao [29] proposed a pointer-based discrete differential evolution (PDDE) for problems where the solutions are expressed as a permutation of integers. They defined addition and multiplication operations based on the concept of memory pointers in some programming languages and used them in the genetic operators of the algorithm. We adopt the PDDE algorithm to solve our problem, as the solution can be expressed as a permutation.

A. Individual Representation

We represent a solution using a permutation of integers, 1, 2, …, *N*+*m*-1. Here, integers 1, 2, …, *N* represent the *N* tasks. The positions of the *m*-1 largest numbers *N*+1, …, *N*+*m*-1 in the sequence separate the tasks into *m* subsequences. The first subsequence is used as the sequence of tasks performed by crane 1, the second subsequence is used as the sequence of tasks performed by crane 2, and so on. Fig. 3 shows an example solution code for a problem with 13 tasks (*N*=13) and 3 cranes (*m*=3). As seen from the figure, the two largest numbers (14 and 15) separate the whole sequence into three subsequences, which indicate that crane 1 performs tasks 9, 3, 8, 11; crane 2 performs tasks 12, 7, 2, 13; and crane 3 performs tasks 6, 10, 4, 5, 1.

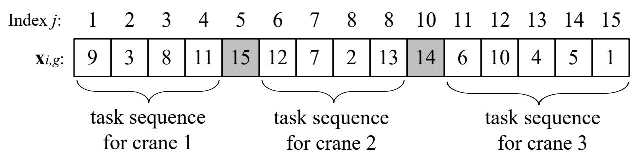
Index j: 1 2 3 4 5 6 7 8 8 10 11 12 13 14 15
x_{i,g}: 9 3 8 11 15 12 7 2 13 14 6 10 4 5 1
task sequence for crane 1
task sequence for crane 2
task sequence for crane 3

Fig. 3. A 15-dimensional individual in PDDE.

For each solution represented by a sequence, we need to calculate the objective function value of the corresponding schedule. This objective value will be used as the fitness function. In the following subsections (B, C and D), we first introduce the operators of mutation, crossover and selection. Then we provide a method for scheduling the tasks and calculating the objective value based on the sequence. Finally, the overall PDDE procedure is presented.

B. Discrete Mutation, Discrete Crossover and Selection

The mutation, crossover and selection operators in this PDDE procedure are defined as in Dong and Zhao [29]. The DE mutation operation generates a mutant individual $\mathbf{v}_i$ for each current individual $\mathbf{x}_i$. The mutation operation of PDDE is a combination of addition and subtraction operations of individuals as shown in (72).

$$\mathbf{v}_i = \mathbf{x}_{r1} \oplus F \odot (\mathbf{x}_{r2} \ominus \mathbf{x}_{r3}) \quad (72)$$

The difference vector $\mathbf{d}_i = \mathbf{x}_{r2} \ominus \mathbf{x}_{r3}$ is obtained by using subtraction operations as illustrated in Fig. 4(a), and the new mutant $\mathbf{v}_i = \mathbf{x}_{r1} \oplus \mathbf{d}_i$ is obtained through addition operations as illustrated in Fig. 4(b). The mutation factor F determines whether the current individual is mutated, as shown in (73). If a random real number in the interval of (0, 1) is less than F, the new mutant $\mathbf{v}_i$ is $\mathbf{x}_{r1}\oplus\mathbf{d}_i$, otherwise it is the base individual $\mathbf{x}_{r1}$.

$$\mathbf{x}_{r1} \oplus F \odot \mathbf{d}_i = \begin{cases} \mathbf{x}_{r1} \oplus \mathbf{d}_i, & \text{if}(rand_i(0,1) < F) \\ \mathbf{x}_{r1}, & \text{otherwise} \end{cases} \quad (73)$$

Single-point crossover operation is then applied to combine the mutation individual $\mathbf{v}_i$ and the target individual $\mathbf{x}_i$. First, a random integer is selected from the interval (1, N+m-1) as the intersection point. Next, the first segment of one of these two individuals and the second segment of the other individual are connected to constitute a preliminary child individual $\mathbf{u'}_i$. Finally, the second segment of this preliminary individual is checked, and the missing elements are added replacing the repeated elements to obtain a feasible child individual $\mathbf{u}_i$. We use $\otimes$ as the crossover operation symbol. Then the situations shown in Fig. 5(a) and Fig. 5(b) can be expressed as $\mathbf{u}_i$=$\mathbf{x}_i\otimes\mathbf{v}_i$ and $\mathbf{u}_i$=$\mathbf{v}_i\otimes\mathbf{x}_i$. With the crossover rate CR, the crossover operator in PDDE is defined as in (74).

$$\mathbf{u}_i = \begin{cases} \mathbf{x}_i \otimes \mathbf{v}_i, & \text{if}(rand_i(0,1) < CR) \\ \mathbf{v}_i \otimes \mathbf{x}_i, & \text{otherwise} \end{cases} \quad (74)$$

C. Fitness Function Calculation

According to the coding method above, for a given sequence, the assignment of tasks and the order for each crane to perform its tasks are determined. However, we still need to work out an overall schedule and calculate the objective value. This is not a straightforward problem. On one hand, we need to decide which tasks should be combined as double-load operations so that the operation time can be minimized. On the other hand, we need to avoid conflict between cranes to ensure the feasibility of the

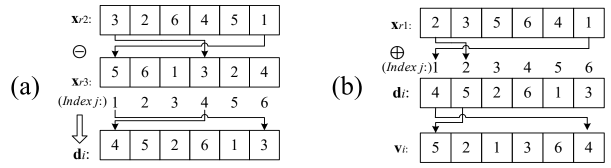

Fig. 4. Examples of subtraction and addition of six dimensions: (a) Discrete subtraction, (b) Discrete addition.

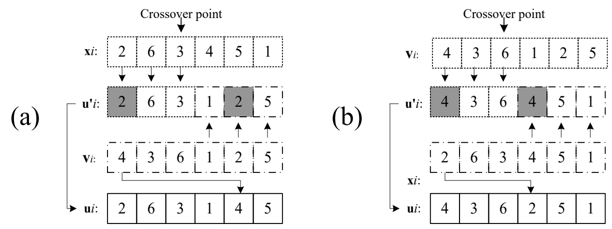

Fig. 5. Examples of discrete one-point crossover operations of six dimensions: (a) $\mathbf{u}_i$=$\mathbf{x}_i\otimes\mathbf{v}_i$, (b) $\mathbf{u}_i$=$\mathbf{v}_i\otimes\mathbf{x}_i$.

schedule. The objective value of each individual solution must be calculated in the DE search process, and so the calculation has to be quick. Therefore, we address the above two issues separately in two steps, first working out a schedule for each crane and then checking the schedules to resolve conflicts, to obtain a feasible overall schedule and the objective value.

Given the subsequence of tasks for one crane, we first check whether any precedence requirements are violated. If the precedence requirement of two tasks is violated, then the order of the two tasks is changed. This is done until all the precedence requirements are satisfied. For the modified subsequence, we develop a forward dynamic programming (DP) procedure to efficiently generate the schedule for the crane to perform the tasks. In each stage of the DP, one of the tasks is considered. We denote the total number of tasks in the sequence as n, and use [i] to represent the i^{th} task in the sequence. There are two states in each stage i: (1) performing task [i] as a single-load operation, and (2) combining it with task [i-1] in a double-load operation. Note that if task [i-1] is combined with task [i-2], then tasks [i] and [i-1] cannot be combined. The links between the states in different stages can be expressed as a network in Fig. 6.

Let $F_i(s)$ be the objective function of state s in stage i representing the time for completion of all tasks up to this state. To simplify the presentation so that stage 2 need not be treated as a special case, we define $o_{[0]-} = o_{[2]+}$. Then the DP can be presented as follows.

First stage:

$$F_1(1) = \frac{|o_{1+} - o_{1-}|}{v}$$

$$F_1(2) = \infty$$

Recursion for other stages:

$$F_i(1) = \min\left\{ F_{i-1}(1) + \frac{|o_{[i-1]-} - o_{[i]+}|}{v^-}, F_{i-1}(2) + \frac{|o_{[i-2]-} - o_{[i]+}|}{v^-} \right\} + |o_{[i]+} - o_{[i]-}| / v$$

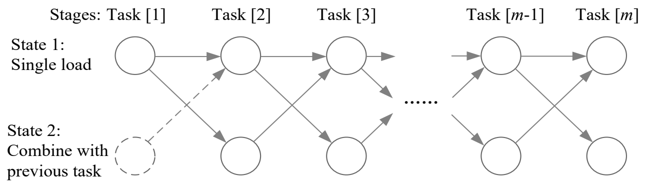

Fig. 6. The network of DP procedure.

$$F_i(2)=\begin{cases} F_{i-1}(1)-\dfrac{\left|o_{[i-1]+}-o_{[i-1]-}\right|}{v}+\dfrac{\left|o_{[i-1]+}-o_{[i]+}\right|}{v}+\dfrac{\left|o_{[i]+}-o_{[i]-}\right|}{v^{+}} \\ \quad +\dfrac{\left|o_{[i]-}-o_{[i-1]-}\right|}{v}, \qquad \text{if } q_{[i-1][i]}=1, \\ \infty,\ \text{otherwise.} \end{cases}$$

Finally, min $\{F_n(1), F_n(2)\}$ will be the optimal objective value for the complete solution. The corresponding schedule for this crane can also be worked out backwards.

The schedules of the cranes are then put together to determine whether they are feasible and, if not, to resolve any conflict. This is done by checking the crane positions at each time point. The cranes cannot cross each other, and they must maintain at least the safety distance at all times. In case of conflict between any two cranes, the operation of one of them will be delayed to avoid the conflict. To decide which crane's operation should be delayed, the tasks they are performing at that time are considered. If the two tasks have a precedence relationship, then the crane performing the task with lower priority will give way to the other. Otherwise, the crane carrying less load will give way. In case the cranes carry the same load, the crane with earlier final completion time will give way. Fig. 7 shows a conflict between cranes 1 and 2 as well as the resolution by delaying the operation of crane 1. While the rules used in this checking process eliminate conflicts, the resulting makespan may not be optimal even if the task assignment, combination and sequencing decisions are optimal. Nevertheless, as will be seen from the experimental results described in section V, the solution obtained is close to optimal. In addition, the rules such as giving way to the crane with heavier load, lead to solutions with good operational safety.

In this way, an overall schedule can be obtained. Based on the task start and finish times in the schedule, we can conduct a final check of the precedence requirements between tasks of different cranes. If they are all satisfied, the schedule is feasible and the makespan of the schedule is taken as the fitness function value. Otherwise, a large penalty value is imposed on this infeasible solution as its fitness function value.

D. PDDE Procedure

With the integer coding and the above method for calculating the fitness values, the PDDE framework can be followed to solve our problem. Algorithm 1 summarizes the overall PDDE procedure used.

Algorithm 1: The PDDE procedure

```
Initialization: Set the maximum number of iterations $g_{\max}$; Let the initial iteration index $g = 0$; Randomly generate an initial population of $NP$ individuals $\mathbf{P}_g=\{\mathbf{x}_{1,g},\ldots,\mathbf{x}_{NP,g}\}$; Calculate the fitness value $f(\mathbf{x}_{i,g})$ of each individual $\mathbf{x}_{i,g}$.
WHILE $g < g_{\max}$
    FOR each individual in $\mathbf{P}_g$
        Generate a mutant $\mathbf{v}_{i,g}$ using the following mutation operation:
            $\mathbf{v}_{i,g}=\mathbf{x}_{r1,g}\oplus F\odot(\mathbf{x}_{r2,g}\ominus\mathbf{x}_{r3,g})$,
                $r_1,r_2,r_3\in[1,D]$, $i\neq r_1\neq r_2\neq r_3$.
        Generate an offspring $\mathbf{u}_{i,g}$ using the following crossover operation:
            $\mathbf{u}_{i,g}=\begin{cases}\mathbf{x}_{i,g}\otimes\mathbf{v}_{i,g}, & \text{if}(rand_i(0,1)<CR)\\ \mathbf{v}_{i,g}\otimes\mathbf{x}_{i,g}, & \text{otherwise}\end{cases}$
        Calculate the fitness value of $\mathbf{u}_{i,g}$: $f(\mathbf{u}_{i,g})$
        Select the better one of $\mathbf{u}_{i,g}$ and $\mathbf{x}_{i,g}$ to survive into the next generation.
            $\mathbf{x}_{i,g+1}=\begin{cases}\mathbf{u}_{i,g}, & \text{if} f(\mathbf{u}_{i,g})\le f(\mathbf{x}_{i,g})\\ \mathbf{x}_{i,g}, & \text{otherwise}\end{cases}$
    END FOR
    $g = g + 1$;
END WHILE
```

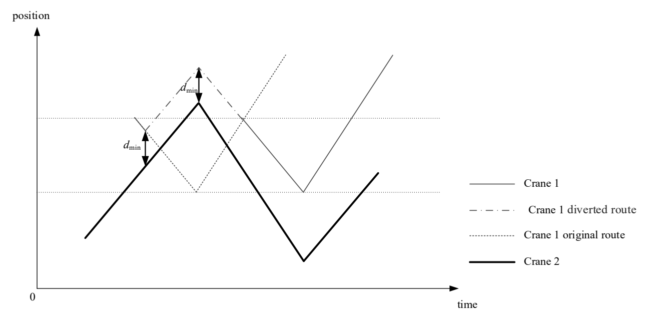

Fig. 7. Illustration of conflict handling.

V. COMPUTATIONAL EXPERIMENTS

Computational experiments are carried out to test the effectiveness and efficiency of the solution methods, using a 64-bit Windows 10 system with an Intel Core i5 2.30 GHz CPU and 8 GB RAM. All the methods are implemented using C++ programming language, and the MILP models are solved using CPLEX 12.51 with default parameter settings.

We used an orthogonal experimental design to tune the PDDE algorithm parameters. The orthogonal experiments were conducted on an 8-task instance taken from actual production. The PDDE has three parameter factors: *NP*, *F*, and *CR*. For each factor, four value levels are chosen to test (see Table I). So, the orthogonal table is selected as $L_{16}(4^3)$ for 16 tests.

The test plan and results are shown in Table II. For each test, we use the average objective value of 10 instances as the response value. Column 'AVG' in Table II show the response values. Row $\overline{\text{AVG1}}$ is the average of the response values at level 1 for each factor. Similarly, $\overline{\text{AVG2}}$, $\overline{\text{AVG3}}$, and $\overline{\text{AVG4}}$ are

TABLE I
PARAMETERS, FACTORS, AND LEVELS

Level	NP	F	CR
1	50	0.1	0.1
2	100	0.3	0.3
3	150	0.5	0.5
4	200	0.7	0.7

TABLE II
ORTHOGONAL TABLE $L_{16}(4^3)$ AND RESPONSE VALUES

Index	NP	F	CR	AVG
1	1(50)	1(0.1)	1(0.1)	1206
2	1(50)	2(0.3)	2(0.3)	1253
3	1(50)	3(0.5)	3(0.5)	1019
4	1(50)	4(0.7)	4(0.7)	1193
5	2(100)	1(0.1)	2(0.3)	988
6	2(100)	2(0.3)	1(0.1)	1289
7	2(100)	3(0.5)	4(0.7)	1107
8	2(100)	4(0.7)	3(0.5)	1215
9	3(150)	1(0.1)	3(0.5)	1097
10	3(150)	2(0.3)	4(0.7)	1241
11	3(150)	3(0.5)	1(0.1)	959
12	3(150)	4(0.7)	2(0.3)	1067
13	4(200)	1(0.1)	4(0.7)	996
14	4(200)	2(0.3)	3(0.5)	1285
15	4(200)	3(0.5)	2(0.3)	1149
16	4(200)	4(0.7)	1(0.1)	1141
$\overline{\text{AVG1}}$	1150	1122	1106	
$\overline{\text{AVG2}}$	**1127**	1167	1164	
$\overline{\text{AVG3}}$	1165	**1065**	**1104**	
$\overline{\text{AVG4}}$	1135	1154	1134	

averages at other levels . The best performance level for each factor is highlighted in Table II, and the parameters used in PDDE are set accordingly as *NP* = 100, *CR* = 0.5, *F* = 0.5. Fig. 8 shows the evolution of the best objective value up to the current generation of the PDDE solution process for one instance. It can be seen that the evolution converges before 5000 generations. This is also observed for other instances and with different parameter settings. Based on this we set $g_{\max}$ = 5000.

Instances of actual problems of different sizes were collected from the slab yard of a steel company. Each instance corresponds to a crane-scheduling scheme in the production plan. The plan contained the initial position and target position of the slab of each task, the precedence relationships between the tasks, and whether the tasks meet the double-load operation conditions. The actual slab yard is equipped with two cranes that are used to perform all tasks in practice. For consistency and fair comparison, two cranes are used for each instance in our numerical experiments. The overall MILP model, the two-phase heuristic, and the PDDE heuristic were used to solve each of the instances. The actual production schedules that were used for these instances were also obtained for comparison. According to the actual production schedule of each instance, we obtain the objective value by using the calculation method in part C of section 4 as a benchmark to compare the performances of the solution methods.

We also proposed a lower bound of the problem based on the

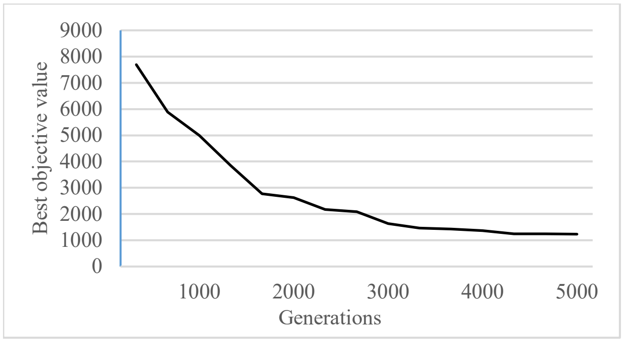

Fig. 8. Evolution of the best objective value up to different generations

first phase of the heuristic method in section III. Considering any optimal schedule of the problem with *m* cranes, we can obtain the route of each crane in it. Linking these routes by adding (*m*-1) empty moves, a feasible single-crane solution to perform all tasks can be obtained. The makespan of this solution should not be shorter than the optimal makespan of the single-crane problem. The above analysis leads to a method of calculating a lower bound of our problem. First, solve the first-phase problem in the two-phase heuristic method to obtain the optimal makespan of the single-crane problem. Then, from this makespan subtract the longest (*m*-1) empty move times between the target stack of a task and the initial stack of a different task; Finally, dividing the result by *m* provides us a lower bound of the multi-crane problem.

The maximum computation time allowed for solving an instance is 4 hours. Within this time limit, the overall MILP model solved instances with up to 9 tasks, and the two-phase heuristic solved instances with up to 11 tasks. We first compare the performances of different methods for these small instances.

For each instance, the objective value of the schedule in practice is used as the benchmark. The percentage improvement made by a tested method over this benchmark is then calculated. The experiment results also show the gaps of the proposed methods to the lower bound. The formulas are as follows.

Improvement = (benchmark – objective value of the test method)/benchmark×100%.

Gap = (objective value of the test method – lower bound)/lower bound×100%.

Table III shows the Improvement, the Gap and Computation time (in CPU seconds) for each method. From Table III, it can be seen that the overall MILP model achieved optimal solutions to the first 16 instances, with average improvement of 24.83%. This indicates that there is great potential for improving the present, manually constructed schedule. For the same 16 instances, the average improvements made by the two-phase heuristic and the PDDE heuristic are 20.93% and 22.50%, respectively. Neither is far from the optimal solution, with the PDDE solution being better than the two-phase heuristic solution. These can also be seen from their gaps to the lower bound. While the average gap of the optimal solution to the lower bound is 9.16%, those for the two-phase heuristic and PDDE are 14.82% and 12.55%, respectively. For all 24 instances in the table, the average improvements of the two-phase heuristic and the PDDE heuristic are 22.79% and 24.23%, respectively, which are slightly higher than those for

TABLE III
EXPERIMENTAL RESULTS COMPARING DIFFERENT SOLUTION METHODS FOR SMALL-SIZED INSTANCES

Instance index	Size	Improvement			Gap			Computation time (s)		
		MILP	2-PHASE	PDDE	MILP	2-PHASE	PDDE	MILP	2-PHASE	PDDE
1	6	21.31%	17.38%	19.77%	9.44%	14.91%	11.59%	48.88	37.27	29.90
2	6	25.66%	21.87%	23.35%	8.97%	14.53%	12.36%	82.28	58.32	35.87
3	6	18.83%	14.61%	16.49%	10.77%	16.53%	13.97%	55.90	41.62	30.34
4	6	22.01%	18.24%	21.43%	9.02%	14.29%	9.83%	62.44	45.74	29.52
5	7	28.29%	24.81%	24.89%	11.51%	16.92%	16.79%	590.94	268.70	38.95
6	7	16.67%	12.38%	13.43%	8.12%	13.69%	12.33%	391.55	242.33	43.69
7	7	26.34%	22.04%	21.65%	9.37%	15.75%	16.33%	433.76	209.16	47.29
8	7	23.62%	19.10%	21.60%	9.75%	16.24%	12.64%	482.58	261.07	46.12
9	8	28.05%	23.78%	25.84%	10.72%	17.29%	14.12%	6339.43	3239.41	58.24
10	8	24.19%	19.56%	22.29%	11.92%	18.76%	14.73%	7057.20	2903.03	46.17
11	8	27.65%	23.27%	26.19%	7.45%	13.95%	9.62%	5728.38	2191.31	52.22
12	8	26.58%	22.15%	24.10%	8.25%	14.78%	11.91%	6859.01	2777.92	59.33
13	9	29.72%	26.92%	27.87%	10.60%	15.01%	13.52%	7041.94	2944.45	76.17
14	9	23.01%	19.87%	19.31%	7.07%	11.44%	12.22%	7187.33	3830.13	84.24
15	9	28.03%	25.14%	25.88%	9.33%	13.72%	12.59%	6843.53	3313.16	90.01
16	9	27.29%	23.78%	25.84%	4.21%	9.24%	6.29%	11383.26	2244.74	81.45
17	10	-	19.16%	20.94%	-	12.97%	10.48%	-	5642.76	113.60
18	10	-	31.05%	31.46%	-	12.41%	11.74%	-	6549.09	99.73
19	10	-	26.25%	28.97%	-	12.03%	7.90%	-	5831.43	117.43
20	10	-	23.14%	25.29%	-	10.26%	7.17%	-	6251.47	99.03
21	11	-	22.87%	22.39%	-	8.74%	9.42%	-	11491.28	106.28
22	11	-	26.96%	28.51%	-	9.33%	7.01%	-	14400	117.93
23	11	-	19.14%	20.94%	-	14.57%	12.02%	-	9388.87	108.40
24	11	-	43.50%	43.14%	-	9.76%	10.45%	-	9655.22	113.39
AVG (Instances 1-16)		24.83%	20.93%	22.50%	9.16%	14.82%	12.55%			
AVG (all 24 instances)		-	22.79%	24.23%	-	13.63%	11.54%			

the first 16 instances. Part of these increases is due to one exceptionally high improvement in instance 24. Overall, the performances of the heuristic solutions appear quite stable.

In terms of computation time, MILP spends more than 3 hours to solve one of the 9-task instances and cannot find a feasible solution for any instances larger than that within 4 hours. The two-phase heuristic reached the 4-hour limit when solving one of the instances with 11 tasks. The PDDE algorithm solved any of the instances within 2 minutes, demonstrating that the PDDE algorithm could solve larger problems in reasonable time. Therefore, we tested it on large practical instances ranging from 20 to 60 tasks. The improvements of the PDDE solution over the current benchmark schedule, the gaps between the PDDE solution and the lower bound and computation times for these large instances are shown in Table IV. It can be seen that the computation time of the PDDE algorithm is only around 15 minutes even for the large problem instances with 60 tasks. PDDE required only a few minutes to solve most of the common instances seen in the company's real operations, which comprise 20 to 30 tasks. Such solution time are suitable for practical use. For these large problems, the average improvements of the PDDE solution over the current benchmark schedule are also stable and similar to those for small problems. The average improvement of the PDDE solution over the actual production schedule is 23.58% and its average gap to the lower bound is 9.44%. These are similar to the averages for small instance. The gap is actually smaller which may be because the lower bound in better for large instances. We also calculated another performance measure, namely the average operation time for each task, which is the makespan divided by the number of tasks completed. These are shown in the last two columns in Table IV. The overall average task operation time of the PDDE solution for all the test large instances is 135 s. For the first 16 small instances, the average task operation times of the optimal solution and the PDDE solution are 129.7s and 134.4s respectively. This indicates that the quality of the PDDE solution for large instances is similar to that for small instances and is close to optimal.

VI. CONCLUSIONS

In this paper, we have studied a multiple double-load crane scheduling problem in steel slab yards. The double-load crane can handle two slabs simultaneously and so requires additional decisions on combinations of tasks for double-load operations. Scheduling multiple cranes on a common track needs to consider non-crossing and safety distance constraints. Each of these features makes the problem more complex. We are not aware of any previous research on crane-scheduling problems that combines both of these features. We first formulated the problem as a mixed-integer linear programming model. Then a two-phase heuristic was proposed with each phase formulated as a smaller MILP model. A pointer-based discrete differential evolution algorithm was then developed with a dynamic programming algorithm embedded to solve the one-crane sub-problem for a fixed sequence of tasks. Computational

TABLE IV
EXPERIMENTAL PDDE RESULTS FOR LARGE SIZE INSTANCES

Instance index	Size	Improvement	Gap	Computation Time (s)	Average operation time for one task	Average operation time of each size for one task
1	20	24.50%	10.54%	216.14	123	134
2	20	24.35%	16.30%	224.91	139	
3	20	23.91%	10.77%	213.57	137	
4	20	28.02%	17.65%	203.30	137	
5	30	20.31%	10.29%	362.36	153	145
6	30	23.10%	9.13%	375.03	143	
7	30	26.77%	8.66%	369.96	141	
8	30	22.81%	12.43%	337.75	143	
9	40	21.71%	8.75%	602.21	135	133
10	40	21.71%	8.00%	611.39	158	
11	40	20.24%	11.46%	662.18	127	
12	40	28.54%	5.77%	608.94	110	
13	50	28.83%	8.71%	718.82	113	129
14	50	21.05%	7.49%	693.94	120	
15	50	24.94%	3.87%	789.83	137	
16	50	21.05%	10.03%	677.83	146	
17	60	22.22%	9.53%	912.00	138	136
18	60	24.28%	3.51%	860.70	129	
19	60	21.05%	7.29%	885.40	135	
20	60	22.15%	8.60%	1001.30	141	
AVG		23.58%	9.44%		135	

experiments were conducted on real instances collected from a steel company, to test the performance of the solution methods. The results show that the MILP solutions greatly improve upon the schedules used in practice, with an average improvement of about 24.83%, although it can only solve small test instances. The qualities of the two-phase heuristic and the PDDE solutions are both near-optimal, with the PDDE solution performing better. In addition, while the problem size that can be solved by two-phase heuristic is still relatively small, the PDDE algorithm solves large practical problems in acceptable time.

REFERENCES

[1] G. Zhao, J. Liu, and Y. Dong, "Scheduling the operations of a double-load crane in slab yards," *Int. J. Prod. Res.* 2019. In press, available online. DOI: 10.1080/00207543.2019.1629666. [Online].
[2] R. Lieberman, and I. Turksen, "Crane scheduling problems," *AIIE Trans.*, vol. 13, no. 4, pp. 304–311, 1981.
[3] A. Lim, B. Rodrigues, F. Xiao, and Y. Zhu, "Crane scheduling with spatial constraints," *Nav. Res. Log.*, vol. 51, no. 3, pp. 386–406, 2004.
[4] W. Ng, "Crane scheduling in container yards with inter-crane interference," *Eur. J. Oper. Res.*, vol. 164, no. 1, pp. 64–78, 2005.
[5] W. Li, Y. Wu, M. Petering, M. Goh, and R. Souza, "Discrete time model and algorithms for container yard crane scheduling," *Eur. J. Oper. Res.*, vol. 198, pp. 165–172, 2009.
[6] W. Li, M. Goh, Y. Wu, M. Petering, R. Souza, and Y. Wu, "A continuous time model for multiple yard crane scheduling with last minute job arrivals," *Int. J. Prod. Econ.*, vol. 136, pp. 332–343, 2012.
[7] J. Liu, Y. Wan, and L. Wang, "Quay crane scheduling at container terminals to minimize the maximum relative tardiness of vessel departures," *Nav. Res. Log.*, vol. 53, no. 1, pp. 60–74, 2006.
[8] D. Lee, J. Chen, and J. Cao, "Quay crane scheduling for an indented berth," *Eng. Optim.*, vol. 43, no. 9, pp. 985–998, 2011.
[9] J. Chen, D. Lee, and M. Goh, "An effective mathematical formulation for the unidirectional cluster-based quay crane scheduling problem," *Eur. J. Oper. Res.*, vol. 232, pp. 198–208, 2014.
[10] G. Alsoufi, X. Yang, and A. Salhi, "Combined quay crane assignment and quay crane scheduling with crane inter-vessel movement and non-interference constraints," *J. Oper. Res. Soc.*, vol. 69, no. 4, pp. 1–12, 2016.
[11] R. Zhang, Z. Jin, Y. Ma, and W. Luan, "Optimization for two-stage double-cycle operations in container terminals," *Comput. Ind. Eng.*, vol. 83, pp. 316–326, 2015.
[12] P. Guo, W. Cheng, Y. Wang, and N. Boysen, "Gantry crane scheduling in intermodal rail-road container terminals," *Int. J. Prod. Res.*, vol. 56, no. 16, pp. 5419–5436, 2018.
[13] F. Zheng, X. Man, F. Chu, M. Liu, and C. Chu, "A two-stage stochastic programming for single yard crane scheduling with uncertain release times of retrieval tasks," *Int. J. Prod. Res.*, 2018. Published. DOI: 10.1080/00207543.2018.1516903. [Online].
[14] P. Hirsch, A. Palfi, and M. Gronalt, "Solving a time constrained two-crane routing problem for material handling with an ant colony optimisation approach: an application in the roof-tile industry," *Int. J. Prod. Res.*, vol. 50, no. 20, pp. 6005–6021, 2012.
[15] L. Lei and T. Wang, "The minimum common-cycle algorithm for cyclic scheduling of two material handling hoists with time window constraints," *Manage. Sci.*, vol. 37, pp. 1629–1639, 1991.
[16] A. Che and C. Chu, "Single-track multi-hoist scheduling problem: a collision-free resolution based on a branch-and-bound approach," *Int. J. Prod. Res.*, vol. 42, pp. 2435–2456, 2004.
[17] Z. Zhou and J. Liu, "A heuristic algorithm for the two-hoist cyclic scheduling problem with overlapping hoist coverage ranges," *IIE Trans.*, vol. 40, no. 8, pp. 782–794, 2008.
[18] J. Leung and G. Zhang, "Optimal cyclic scheduling for printed circuit board production lines with multiple hoists and general processing sequence," *IEEE Trans. Robot.*, vol. 19, pp. 480–484, 2003.
[19] J. Leung, G. Zhang, X. Yang, R. Mak, and K. Lam, "Optimal cyclic multi-hoist scheduling: a mixed integer programming approach," *Oper. Res.*, vol. 52, pp. 965–976, 2004.
[20] J. Liu and Y. Jiang, "An efficient optimal solution to the two-hoist no-wait cyclic scheduling problem," *Oper. Res.*, vol. 53, pp. 313–327, 2005.
[21] Y. Jiang and J. Liu, "Multi-hoist cyclic scheduling with fixed processing and transfer times," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, pp. 435–450, 2007.
[22] Y. Jiang and J. Liu, "A new model and an efficient branch-and-bound solution for cyclic multi-hoist scheduling," *IIE Trans.*, vol. 46, no. 3, pp. 249–262, 2014.
[23] A. Dohn and J. Clausen, "Optimising the Slab Yard Planning and Crane Scheduling Problem using a two-stage heuristic," *Int. J. Prod. Res.*, vol. 48, no. 15, pp. 4585–4608, 2010.
[24] X. Xie, Y. Zheng, L. Tang, and Y. Li, "Multiple crane scheduling in a batch annealing process with no-delay constraints for machine unloading," *Appl. Math. Model.*, vol. 49, pp. 470–486, 2017.

[25] G. Maschietto, Y. Ouazene, M. Ravetti, M. de Souza, and F. Yalaoui, "Crane scheduling problem with non-interference constraints in a steel coil distribution centre," *Int. J. Prod. Res.*, vol. 55, no. 6, pp. 1607–1622, 2017.

[26] C. Zhou, H. Li, W. Wang, L. H. Lee, and E. P. Chew, "Connecting the belt and road through sea-rail collaboration," *Frontiers Eng. Manage.*, vol. 4, no. 3, pp. 315–324, 2017.

[27] L. Tang, Y. Zhao, and J. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 209–225, 2014.

[28] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, 2015.

[29] Y. Dong and R. Zhao, "Solve train stowage planning problem of steel coil using a pointer-based discrete differential evolution," *Int. J. Prod. Res.*, vol. 56, no. 22, pp. 6937–6955, 2017.

Guodong Zhao holds a B. Eng. degree in Automation and a M. Eng. degree in Systems Engineering from Northeastern University, Shenyang, China (awarded 2009 and 2011 respectively). He is currently working toward a PhD in Systems Engineering at the Institute of Industrial and Systems Engineering, Northeastern University, Shenyang, China.

Mr. Zhao's research interests include production planning and scheduling, modeling and optimization in the steel industry, decision support systems, and metaheuristics-based optimization algorithms. His research paper has been published in academic journal *International Journal of Production Research*.

Jiyin Liu is Professor of Operations Management at the School of Business and Economics at Loughborough University, UK. He previously taught at Northeastern University and the Hong Kong University of Science and Technology in China. He received his PhD in Manufacturing Engineering and Operations Management from the University of Nottingham in the UK, and his B. Eng. in Industrial Automation and M. Eng. in Systems Engineering from Northeastern University, China.

Prof. Liu's research interests are in the areas of operations planning and scheduling in production and logistics, as well as in modeling, analysis, and solution of practical operations problems. His research has been published in various academic journals, including *European Journal of Operational Research, IEEE Transactions, IIE Transactions, International Journal of Production Research; Manufacturing & Service Operations Management, Naval Research Logistics, Operations Research, and Transportation Research*.

Lixin Tang (M'09–SM'14) received a B. Eng. degree in Industrial Automation, M. Eng. degree in Systems Engineering, and PhD in Control Theory and Application from Northeastern University, Shenyang, China (1988, 1991, and 1996, respectively).

He is a Cheung Kong Scholars Chair Professor, and Director of the Institute of Industrial & Systems Engineering at Northeastern University, China.

Prof. Tang's research interests cover industrial big-data science, data analytics and machine learning, reinforcement learning and dynamic optimization, computational intelligent optimization, plant-wide production and logistics planning, production and logistics batching and scheduling and engineering applications in manufacturing (steel, petrochemical, nonferrous), energy, resources industry and logistics systems. His research papers have appeared in academic journals such as *Operations Research*, *IIE Transactions*, *Naval Research Logistics*, *IEEE Transactions on Evolutionary Computation*, *IEEE Transactions on Power Systems*, *IEEE Transactions on Control Systems Technolog*y, and the *European Journal of Operational Research*.

Prof. Tang serves as an Associate Editor of *IISE Transactions, IEEE Transactions on Evolutionary Computation*, *IEEE Transactions on Cybernetics*, *Journal of Scheduling*, *International Journal of Production Research*, *Journal of the Operational Research Society*; on the Editorial Board of *Annals of Operations Research*; and is an Area Editor of the *Asia-Pacific Journal of Operational Research*.

Ren Zhao is a postdoctoral faculty member of the Institute of Industrial and Systems Engineering, Northeastern University, Shenyang, China, where she also gained her PhD. Her research interests include production scheduling problems of iron and steel enterprises, logistics scheduling, port container logistics scheduling problems, and optimization theories and methods. As the main researcher of the project, Dr. Zhao participated in the National Science Foundation for Outstanding Youth (70425003), National High-tech Research and Development Plan (2006AA04Z174), and other projects. Her research papers have been published in academic journals such as *Naval Research Logistics* and *Journal of Automation*.

Yun Dong received a Bachelor's degree in automation from Harbin Institute of Technology in 2005 and a Master's degree and PhD in Systems Engineering from Northeastern University, Shenyang, China (2009 and 2015 respectively). He is currently a lecturer at the Institute of Industrial and Systems Engineering, Northeastern University. Dr. Dong's research interests include planning and scheduling, modeling and optimization, meta-heuristics, machine learning, and the development of decision support systems. His research papers have been published in academic journals such as *IEEE Transactions on Evolutionary Computation*, *IISE Transactions*, and *International Journal of Production Research*.